TimeServerVlbi

-

A VLBI data stream real-time monitor

Matteo Stagni

March 25, 2015

IRA 485/15

Abstract

A couple of Python scripts that serve the purpose of monitoring VLBI data streams and check their time coherence. The software has been developed to test and verify in real time data streams coming from a dBBC backend using a Fila10G formatter without the need to prepare a VLBI scheduled experiment.

Contents

1 Introduction

Since the introduction of a new digital backend, the dBBC, in all Italian VLBI antennas facilities there has been an acceleration towards new ways of dealing with VLBI data. The network capabilities provided by the Fila10G formatter, an FPGA board capable of streaming UDP data packets up to 4 Gbit/s has enabled remote recoding on COTS hardware and software, and further pushes towards real-time correlation.

The pitfalls of dealing with this new formatter have spurred us to find a rapid way to test the newly introduced capabilities such as the VDIF data format. Planned and unforeseen maintenance time at the antennas have done the rest.

The overall program was thought as a 'surrogate' of a real-time VLBI experiment which is more time consuming to prepare and requires pointing a source at the antennas. This would have been difficult to schedule and would have required extra work for operators. Instead a *timeServerVlbi* instance can be run even when there is no actual observation in place, but only the formatter is on sending out packets on the network.

2 MARK5B

The Mark5B header format is the *de-facto* present standard when dealing with VLBI data. It is supported by a number of recorders and correlators, though its dimension was not designed with network capabilities in mind. Its biggest downside lays in its frame length, 10000 bytes which cannot be supported by standard network appliances which are limited to Jumbo frames up to a size of 9000 bytes.

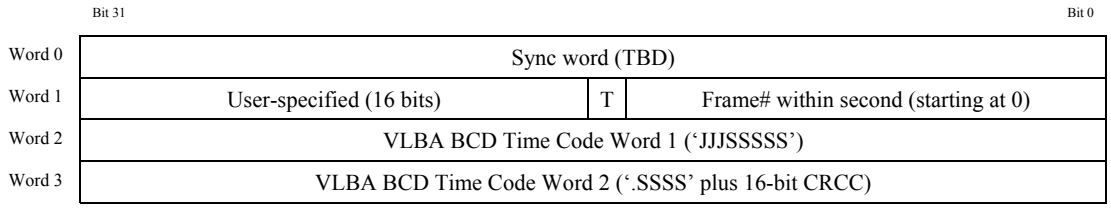

Figure 1: Disk Frame Header format [3]

Looking at the composition of the header we can find a useful sync word at the beginning (*ABADDEED*), followed by the frame number within second, which depends on the sampling rate decided for the observation.

There is a sensitive bit that follows the frame number section in Figure 1 marked as *T* which signals the presence of the *Test Vector Generator*, random noise produced by the formatter to test the hardware. The presence of this bit has proved difficult to manage in recent times because most correlators have been programmed to discard such data, whereas it is occasionally included into the frame number section like when the sampling rate of the VLBI experiment rises over 2 Gbit/s.

The last two words of the Mark5B header are composed by a 4 bytes time code where the first three numbers represent a shortened Modified Julian Date day (*JJJ*) and the last five numbers the second of the day starting from 0 at midnight (*SSSSS*). The following word contains the fractional part of the second (*.SSSS*)[1]and 2 bytes of CRCC (usually marked as zeros).

2.1 MARK5B data format on the network

In order to transmit Mark5B data packets on a TCP/IP network in UDP mode, a Fila10G board splits the 10000 bytes Mark5B frame into two 5008 UDP packets, preceded by a Packet Serial Number (*PSN*). The serial number is identical for the two packets, though only the first one contains the header. It is usually the task of the recorder program to discard single packets or in case of a successful transmission, to join them together and write the reconstructed frame to disk.

This design complicates the analysis of the header, in this case the sync word in the Mark5B header helps to determine whether the packet contains a valid header or not.

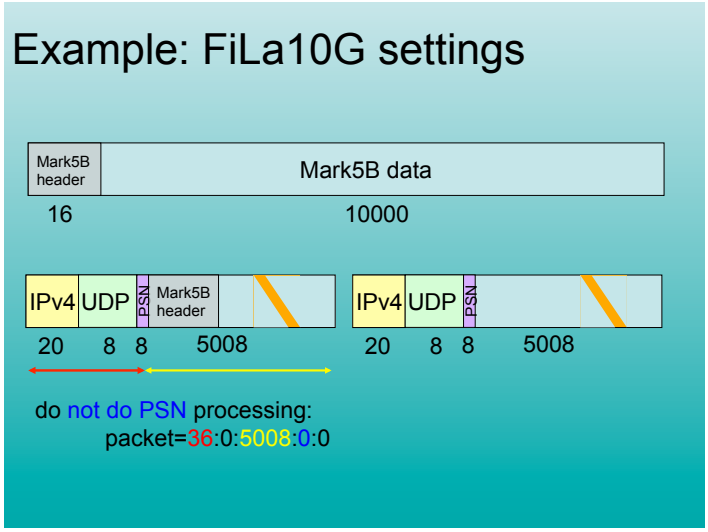

Figure 2: Mark5B network data packets as produced by a Fila10G formatter board

[1]When using a Fila10G formatter, in anticipation of the VDIF design, the fractional part of the second is absent, usually marked as zeros. It is thought to be sufficient to use the frame counter to determine the fractional part of the second. This is not that case when the formatter is a Mark5B recorder, which fulfills the complete standard and marks the fractional part of the second according to the connected PPS (Pulse Per Second) from a Maser clock

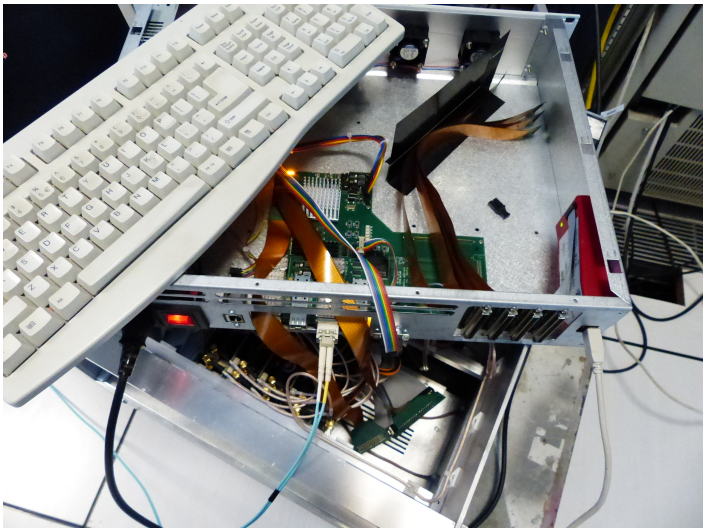

*Figure 3: A Fila10G (**Fi**rst - **la**st) external board*

3 VDIF

	Bit 31 (MSB) Byte 3		Byte 2	Byte 1	Byte 0 Bit 0 (LSB)
Word 0	I_1	L_1	Seconds from reference epoch$_{30}$		
Word 1	Un-assigned$_2$	Ref Epoch$_6$	Data Frame # within second$_{24}$		
Word 2	V_3	$\log_2$(#chns)$_5$	Data Frame length (units of 8 bytes)$_{24}$		
Word 3	C_1	bits/sample-1$_5$	Thread ID$_{10}$	Station ID$_{16}$	
Word 4	EDV$_8$		Extended User Data$_{24}$		
Word 5	Extended User Data$_{32}$				
Word 6	Extended User Data$_{32}$				
Word 7	Extended User Data$_{32}$				

Figure 4: VDIF Data Frame Header format; subscripts are field lengths in bits; byte numbers indicate relative byte address within 32-bit word in little endian format [1]

The VDIF data format has been designed to overcome all the problems previously mentioned. Every packet that is sent to the network, or reordered, contains an header and in Word 2 is defined the data frame length which one would sensibly limit to fit into a Jumbo frame.

Timestamp inside the header has seen an evident change from the Mark5B format. Now there is a 6 bit Reference Epoch counter in Word 1 (*Ref Epoch*) that is incremented every six months starting from year 2000 (0), so the seconds now do not reset at midnight every day, but instead begin from the reference epoch considered. Seconds can be found in Word 0 and even if this design may not be thought as straightforward as the previous one, it prevents leap second problems.

A data frame number is still present to mark the sampling rate of the experiment and determine the fractional part of the second. In this case more room has been allowed (3 bytes) to fulfill the requirements of higher rates.

3.1 VTP - VDIF transport protocol

In addition to this design there are 8 bytes that come before the VDIF header, similar to the Mark5B PSN previously mentioned. When using a Fila10G formatter in recent firmare implementations the VTP 'pre-header' is a counter that starts the moment the Fila10G begins sending data. This may be helpful in case of missing packets or packets that come in wrong ordering due to the stateless UDP transfer protocol.

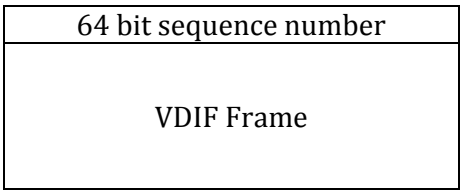

Figure 5: VTP/UDP packet structure [2]

4 timeServerClient

The scripts architecture implemented to analyze VLBI data packets coming from a network stream and verify time coherence between various streams begins on the client side.

timeServerClient is meant to capture the streams coming from the network, produced by a formatter either in Mark5B or VDIF mode . In a Mark5B stream it is necessary to first identify the packets containing headers, then detect whether the header contains *TVG* bit to give out a warning.

This preliminary Mark5B header analysis causes a bit of CPU load on the machine, though it is necessary in order to prevent unwanted interferences by the TVG bit. In fact the MJD + seconds bits of the header may be interpreted a signed or unsigned int whether the *TVG* is present or not.

If the *TVG* bit is detected, then the 2 bytes containing the MJD and seconds are masked and the result is passed on to the timeServerClient UDP sender thread.

In case VDIF packets are received from a formatter, the 3 VDIF bytes representing seconds stripped by it timeServerClient won't need any analysis, but will be simply passed on.

In either case, once the program is started, the user will receive an immediate feedback about the first valid packet captured by *timeServeClient* containing the seconds information, an eventual *TVG* presence warning when in Mark5B mode and the selected IP destination of the packets including port number.

The UDP sender thread is sending the second bytes to a *timeServerVlbi* instance only when the frame number in any kind of header is marked as zero, which means at the beginning of

each second. In this way the workload is balanced across each machine that receives VLBI UDP streams. Multiple *timeServerClient* instances could theoretically be run on the same machine if receiving and sending ports are differentiated, though this is not advisable.

4.1 Usage

```
usage: timeServerClient.py [-h] [-p UDP_PORT] [-S SRV_IP] [-P SRV_PORT] [-v]

timeServerClient is the client part of timeServerVlbi that analyses packets
coming from a formatter.

optional arguments:
  -h, --help            show this help message and exit
  -p UDP_PORT, --port UDP_PORT
                        port listening for formatter datastream
  -S SRV_IP, --severip SRV_IP
                        time server ip
  -P SRV_PORT, --serverport SRV_PORT
                        time server port
  -v, --vdif            VDIF mode
```

5 timeServerVlbi

A *timeServerVlbi* instance could be run on any machine receiving a VLBI data stream , though depending on the experiment sampling could be wise to place it onto a third machine, preferably connected with the other machines receiving data streams through an InfiniBand connection that guarantees low latency.

The script main purpose is to log the seconds received from each macine running *timeServerClient* and produce information whether there are time slips on the formatters or any other kind of useful information that could be extrapolated.

A user can automatically set multiple UDP receiving threads depending on the number of data streams to be confronted. The script has been tested up to three machines. Comparing a high number of data streams could potentially strain threads, though given the present number of VLBI antennas it is at present a remote hypothesis.

The main program thread analyses the VLBI seconds packets received from various *timeServerClient* instances by marking their arrival time at the machine clock precision (microseconds). Two queues are set up for each stream instance, one for the second bytes received and the other for the arrival time of the packet. These queues are then confronted, first if they are the same size, then if there are differences they are emptied. This can occur if there are network problems, like packets getting lost, or disordered. When queues are all the same dimension, and there are still differences in header seconds arriving in *timeServerVlbi* queues, then they are logged as a time slip error from the formatter.

This analysis implies that the formatter streaming source, like a Fila10G board, is close enough to be recorded within a second.

5.1 Usage

```
usage: timeServerVlbi.py [-h] [-i HOST] [-c UDP_IPS] [-p UDP_PORTS] [-l LOG]
                         [-v]

Instructions for timeServer

optional arguments:
  -h, --help            show this help message and exit
  -i HOST, --ip HOST    your host IP
  -c UDP_IPS, --client UDP_IPS
                        add client IPs MIN 2 MAX 4 eg : [-c 192.168.1.10 -c
                        192.168.1.11 ...]
  -p UDP_PORTS, --port UDP_PORTS
                        add client ports MIN 2 MAX 4 eg : [-p 5001 -p 5002
                        ...] make sure they MATCH the order of the IPs
  -l LOG, --log LOG     log path/file name
  -v, --vdif            VDIF mode
```

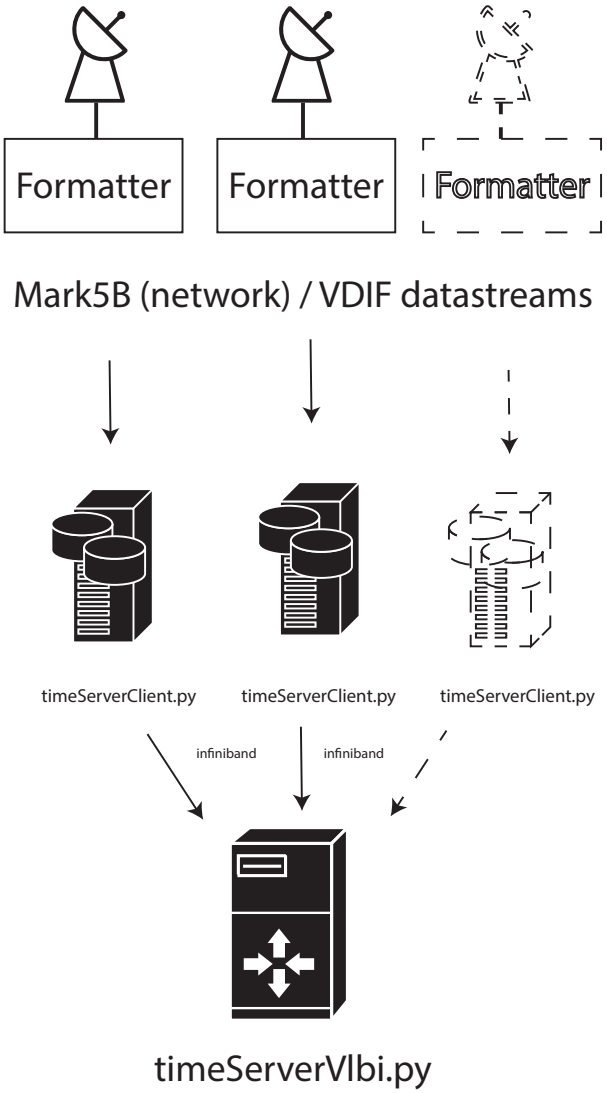

Figure 6: Working schema

6 Conclusions

The scripts development was dictated by the necessity to verify in real time the time synchronization of various Fila10G boards that still at the time of writing this paper need constant checking due to unexpected behavior. Time sync is still a manual task to be performed by operators before experiments, though an introduction of a GPS receiver to automatically synchronize time on the boards may mitigate the problem.

On a second instance scripts have been adapted to be used by a general public, using any kind of formatter as long as it produces a valid VDIF output format. This format is still undergoing minor revisions and adjustments, like the introduction of the VTP counter which was not found on earlier versions of the Fila10G firmware, so the scripts may need to be adapted to future changes of the formatter output.

An interesting side usage of the scripts is to check network latency, as the log file reports the arrival time of packets coming from different sources. For instance we have found that Medicina has a network latency of ~0.0006 seconds and Noto ~0.017 seconds, when receiving packets in Bologna.

The scripts package is going to be integrated into the DiFX distribution and can also be found at this address:

`http://vlbi-mgr.ira.inaf.it/timeserver/TimeServer.tar.gz`

References

[1] Chris Phillips, Alan Whitney, Mamoru Sekido, and Mark Kettenis. Vlbi data interchange format (vdif) specification. Technical Report 1.1.1, MIT Haystack Observatory and JIVE and CSIRO/ATNF and NICT, http://www.vlbi.org/vdif/, June 2009.

[2] Chris Phillips, Alan Whitney, Mamoru Sekido, and Mark Kettenis. Vtp: Vdif transport protocol. Technical Report 1.0.0, http://www.vlbi.org/, October 2013.

[3] Alan Whitney and Roger Cappallo. Mark5 memo 019. Technical Report 19, MIT, HAYSTACK OBSERVATORY, http://www.haystack.mit.edu/, 2004.